The better team always wins - or does it?

Understanding how random chance affects the outcome of an ice hockey game

Daniel Kari

A thesis presented for the degree of
Master of Science.

Department of Mathematics and Statistics
Faculty of Science
University of Helsinki
6.4.2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Mathematics and Statistics

Tekijä — Författare — Author
Daniel Kari

Työn nimi — Arbetets titel — Title

Understanding how random chance affects the outcome of an ice hockey game

Oppiaine — Läroämne — Subject
Statistics

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	April 2020	46

Tiivistelmä — Referat — Abstract

Estimating the effect of random chance ('luck') has long been a question of particular interest in various team sports. In this thesis, we aim to determine the role of luck in a single ice hockey game by building a model to predict the outcome based on the course of events in a game. The obtained prediction accuracy should also to some extent reveal the effect of random chance.

Using the course of events from over 10,000 games, we train feedforward and convolutional neural networks to predict the outcome and final goal differential, which has been proposed as a more informative proxy for outcome. Interestingly, we are not able to obtain distinctively higher accuracy than previous studies, which have focused on predicting the outcome with infomation available before the game. The results suggest that there might exist an upper bound for prediction accuracy even if we knew 'everything' that went on in a game. This further implies that random chance could affect the outcome of a game, although assessing this is difficult, as we do not have a good quantitative metric for luck in the case of single ice hockey game prediction.

Avainsanat — Nyckelord — Keywords
ice hockey, convolutional neural networks, sports analytics

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Acknowledgments

First and foremost I would like to thank my instructor Arto Klami for the guidance in this year-long project. I am particularly grateful for the opportunity to write my thesis on a topic that genuinely interests me. I don't know if I could have managed the workload otherwise.

Contents

Chapter 1

Introduction

The better team *always* wins.

The more I watch ice hockey, or team sports in general, the less I believe the aforementioned sports cliche is true. I cannot tell exactly when I first really began to question it, but one early turning point was perhaps Friday March 18th 2011. In a much anticipated first game of a best-of-seven series, my childhood favorite ice hockey team Jokerit was facing their fierce local rivals HIFK in the Finnish league quarterfinals. I watched from the away stand as HIFK, who had finished three spots higher in the regular-season and thus were the slight favorite, outplayed their visitors, firing 39 shots on goal against Jokerit's 24. Despite this, it was Jokerit who scored the only goal of the game and took the win after a solid defensive effort and a remarkable performance by goaltender Jan Lasak.

I remember witnessing something out of the ordinary that night. HIFK had created more than enough chances to score at least one goal, hitting the goal post twice in the process, while Jokerit had had very limited scoring opportunities. Of course I was beyond thrilled about the surprise win against an arch nemesis, but somehow I knew that if the trend continued, it was unlikely that Jokerit could go on and take the series, even though they managed to steal that first game. To my bitter disappointment, after seven hard fought games HIFK ultimately advanced to the semi-finals. I cannot say that I was terribly surprised about this though, as HIFK stayed on the front foot for most part of the series.

Another defining moment for my skepticism occurred eight years later on June 12th 2019. I was sitting with a couple of friends watching the St. Louis Blues face the Boston Bruins in the seventh and decisive game of the National Hockey League (NHL) finals. Retrospectively, the turning point of the game was already in the first period. After complete domination by the Bruins in every aspect of the game except for goals, the Blues scored two goals in the last

Figure 1.1: The St. Louis Blues celebrated their first ever Stanley Cup in the ultimate game of the 2018-19 NHL season, which is coincidentally also the chronologically last game in the dataset used in this thesis. (Getty Images)

four minutes of the period, with the second one coming only 7.9 seconds before the buzzer. In the remaining two periods Blues were successful in defending their lead and limiting the Bruins scoring chances. Eventually they won the game 4-1 and celebrated Stanley Cup awarded to the playoff champion of the NHL, as illustrated by Figure 1.1.

Still, were the Blues truly the better team in that deciding first period, which they won 2-0, despite going without a single shot on goal for over 15 minutes before their opening goal? I cannot help but think that had the Bruins found a way to get one past Blues netminder Jordan Binnington in the first period when the game was still scoreless, the outcome of the game and thus the whole series could have very well been completely different. This is, of course, purely speculative, as it is not possible to go back in time and play the game again from a certain starting point.

Ultimately, the question of whether the better team always wins or not, boils down to if random chance, or luck, affects the outcome of a game. This is something other ice hockey fans are contemplating as well. In Finnish online ice hockey discussion board Jatkoaika, a thread on the subject [24] has received 189 posts. Almost all of the fans agree that luck plays a role in the game, but the question of whether or not random chance has a significant effect on the outcome, i.e. if it causes the better team to sometimes lose, is more divisive.

How could the effect of random chance be studied appropriately? One possible way could

by inspecting if there are some underlying and repeating patterns in an ice hockey game, which are characteristic to the winning team. If such patterns exist, by capturing them we should be able to predict the winner accurately.

This is exactly what we are doing in this thesis. Our aim is to predict the outcome of an ice hockey game using the *in-game events*, also called *plays*, in that game. In this thesis, we define in-game events as events that occur when the game is going on and are thus related to the 'flow of the game'. In ice hockey, these events are for example shots and hits.

We restrict to using only the in-game events as predictors in order to determine how accurately we can infer the outcome if we knew 'everything' that happened in a game. As a consequence, the results should at least to some extent unveil the effect of random chance on a single game level. The gap between perfect accuracy and the prediction accuracy we are able to reach in this thesis can be interpreted as an upper bound for the amount random chance contributes to the outcome.

In prediction, we choose to estimate two quantities: the outcome itself and the final goal differential, which has been proposed as a more informative proxy for outcome [13]. As every ice hockey game consists of at least three 20 minute periods, but not necessarily more than that, we decide to predict the outcome (home win, tie or away win) and goal differential after 60 minutes for all games. We have access to course of events in over 10,000 ice hockey games [12]. This, along with the fact that except for goals we do not know what exactly determines the winner of a game, makes machine learning suitable approach for the prediction task.

To my best knowledge, this kind of study has not yet been conducted in ice hockey or in other sports either. The approach is fundamentally different to earlier studies presented in Chapter 4, which have focused on predicting the outcome based on information available before a game. These studies generally define the better team as the team considered stronger before a game. As an important distinction, in this thesis we determine the better team only on an individual game level without any pre-existing knowledge about team strength levels.

The structure of the thesis is as follows. Ice hockey as a game and the NHL as a a league are introduced in section 1.1. The essential theoretical background regarding machine learning and neural networks is presented in Chapters 2 and 3. In Chapter 4, we will briefly discuss what type of outcome prediction has been performed across team sports and introduce a theoretical upper bound for prediction accuracy. Chapter 5 focuses on the data used in this thesis, while in Chapter 6 we will go through the models fitted in detail. In Chapter 7 the key findings of the thesis are presented and put into context. Lastly, in Chapter 8 we will discuss the results and present some ideas for future research.

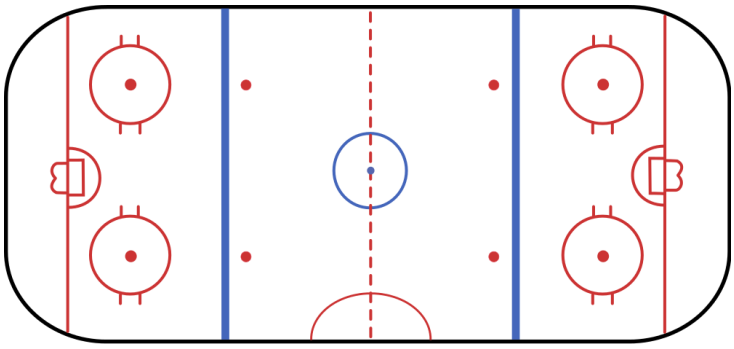

Figure 1.2: Ice hockey rink [10].

1.1 About ice hockey and the NHL

Ice hockey is a team sport in which two teams use sticks to shoot a special rubber disk called puck into opponent's net, also called goal. The purpose of the game is to score more goals than the the other team, as the team which scores the most goals is the winner. Ice hockey is a relatively low-scoring sport, as on average less than six goals are scored per game. Each team typically has five field players and a goaltender on the ice at the same time. Simplifying a little, the field players aim to score goals on the opponent goal, while the goaltender's job is to prevent shots entering his net. Competitive ice hockey is played on a special enclosed area of ice called rink, which is depicted in Figure 1.2. One game consists of three 20 minute periods, between which an intermission is held. The clock stops if the referee blows their whistle, which can for example be caused by a goal scored or infraction of the game's rules, which are extensively covered by the official International Ice Hockey Federation rulebook [20].

National Hockey League (NHL) is the most popular professional ice hockey league in the world. There are currently 31 teams in the league, 7 of which are based in Canada and the remaining 24 in the United States. The teams are divided to two conferences, Eastern and Western, which have 16 and 15 teams respectively. The Eastern Conference is further split into two divisions, the Atlantic and the Metropolitan, which both have 8 teams, while the Western Conference consists of Pacific and Central division with 7 and 8 teams, respectively. Every year regular season of 82 games is played from October through April. After regular season, the 3 top-seeded teams of each division, in addition to remaining two best-seeded teams of each conference, advance to the post-season knockout tournament, also called playoffs. Each playoff series is played with a best-of-seven format, meaning the first team to reach four wins advances to the next round. After four consecutive rounds only one team remains, which is awarded the Stanley Cup as the champion of the NHL.

Chapter 2

Machine learning

Machine learning can be broadly defined as a process in which a machine changes its behavior to improve expected future performance based on past experience [38]. It is commonly used when a problem cannot be solved explicitly or when creating such solution is not feasible. This means that error is an indispensable part of solutions provided with machine learning, except for trivial cases. In all other occasions one needs to settle for 'good enough'. In his book, Mitchell [33] gave more formal definition to machine learning:

Definition 2.1. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

The key difference between machine learning and traditional algorithms is that in machine learning we do not have any pre-defined set of rules. Instead, we, generally under some constraints, let the machine make the decisions. This approach is especially suitable in this thesis. It would be extremely difficult, if not impossible, to come up with a traditional algorithm that could predict the outcome of an ice hockey game accurately. However, with machine learning we can let the machine 'learn' to predict the outcome without needing to understand the underlying decision process completely.

It is a common misconception to assume that machine learning is just a new fancy term for traditional statistical modelling. According to Stewart [47], the two have fundamentally different approaches, even if the methods used can be very similar or even identical. In statistical modelling, the focus is typically on inferring relationships between variables, whereas in machine learning the goal is in most cases to solve the task in question as accurately as possible. With machine learning we generally lose the *interpretability* of the model at least to some extent, meaning that we cannot always tell what exactly makes the machine learning algorithm work the way it does.

For example, in the case of hockey-related analysis we might be interested in determining if face-off win percentage affects the outcome of a game. There are numerous statistical tests and procedures that could be used to examine this. After careful analysis we might be able to conclude whether the effect is positive or not or if the two events are purely coincidental. By contrast, in machine learning we might be interested to use face-off win percentage along other variables to try and predict the outcome of the game. Depending on the model selected, it could be almost impossible to explain the individual effect face off win percentage has on the outcome. In this thesis, we are satisfied with models that can provide sufficiently accurate predictions and thus we do not consider interpretability of our models integral.

2.1 Basic concepts

When machine learning is used in practice, the first step is to specify the task we want to solve. This might for example be predicting the outcome of an ice hockey game. After that suitable data is gathered and possibly labeled manually. The quality and quantity of the data are essential to gain meaningful results [34]. We might also want to determine which variables, which in the context of machine learning are often called *features*, in the data are most relevant for the task in question. This phase might include formal process of *feature selection*, which can be considered as a field of its own. *Feature construction* is the process of adding new features to the data based on existing ones in order to improve predictions.

The learning structure, also called learner or model, then receives subset of the data for *training*. In *supervised learning* the data consists of input-output pairs, such as the course of events and the outcome or final goal differential of a game. Contrary to this, in *unsupervised learning* we do not have any output value and the aim is often to discover structural similarities between the samples. In this chapter, the focus is on supervised learning since it is more relevant to the research question of this thesis. However, since the output values do not tell anything about the effect of random chance, our ultimate goal is in some sense unsupervised.

When the output space is discrete, the task is called *classification*. Determining the winner of an ice hockey game is a good of example of a simple classification problem, since all games result in either home win or away win. By contrast, in *regression* the output is a numerical value, such as the final goal differential. The difference between the tasks affects for example how the quality of prediction is measured.

The most fundamental objective in machine learning is *generalization* [34]. After training, the model should be able to predict the output for unseen data points, also called *test set*, as

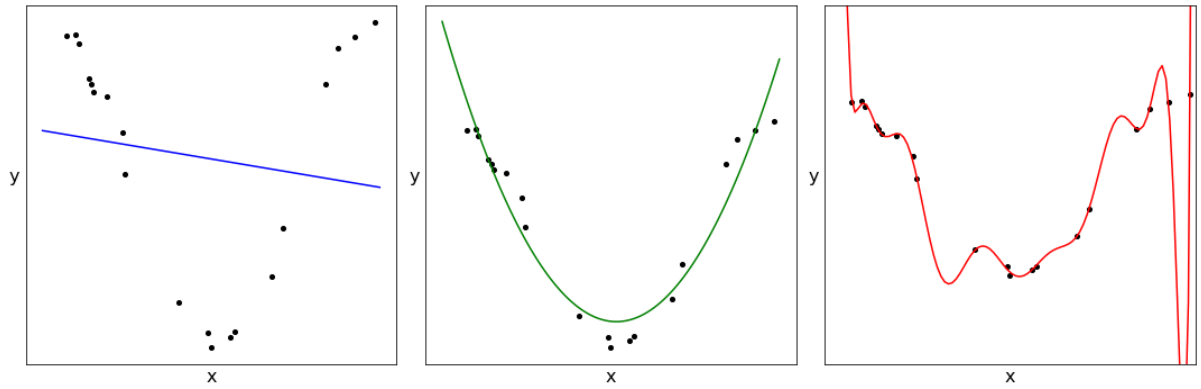

Figure 2.1: The model on the left fails to capture the parabolic pattern in the data and is thus underfitting. The model on the right overfits by adapting to random variation in the data. The model on the middle has optimal balance between bias and variance and is likely to produce the lowest error for new data.

accurately as possible, that is to say generalize well. If the model is very complex or flexible, it might be able to predict the labels for training sample perfectly or with very little error. However, this does not guarantee the prediction will be accurate for different sample of the data. The function is *overfitting*, if it models random noise in the training set, resulting in decreased prediction accuracy for the test data. Correspondingly the model can also be too simple and *underfit*, if it fails to detect patterns present in the data. These phenomena are visualized in Figure 2.1. The process of reducing the risk of overfitting is called *regularization*.

The dilemma between underfitting and overfitting is called *bias-variance tradeoff*. Variance refers to the effect different training data set would have to the model, whereas bias indicates the amount current model is unable to learn because it is too simple [22]. It is trivial to create model with either very low variance or very low bias, but obtaining the optimal balance between the two is more difficult. Typically as the model becomes more flexible, the variance will increase and the bias will decrease.

2.2 Loss function

Loss function is a function that measures the difference between the predicted and the true label [34], which can be interpreted as the cost of a decision. In the training phase of a machine learning process, the goal is to minimize the expected loss, also called *risk*, for the training data. The minimization can be thought to be an optimization problem and thus it is closely related to field of mathematical optimization.

There is no loss function that would be suitable for every situation. The selection of loss function depends e.g. on whether the task is regression or classification and what is the cost of error. Common loss functions include e.g. squared loss, which is used in regression and log-loss, which is good for classification tasks. Both loss functions are easily differentiable and have nice mathematical properties, which contribute to their popularity. Squared loss is defined as

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{2.2}$$

where y_i is the true output for sample i and $\hat{y}_i$ is the predicted output for sample i. Correspondingly, log-loss is defined as

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{n} \sum_{j=1}^{k} \mathbb{1}(y_i = j) \log P(\hat{y}_i = j), \tag{2.3}$$

where k is the number of classes, $\mathbb{1}$ is the indicator function and $P(\hat{y}_i = j)$ is the predicted probability for sample i to belong to class j. Log-loss is often used in classification instead of simpler misclassification rate or 0-1 loss, because it continues to decrease after misclassification rate has reached zero for the training data. This allows building a classifier that is able to separate the classes better [14].

2.3 Model evaluation

The number of possible models that can be fitted to a data is unlimited, since there are myriad different algorithms we can try and for each model we can always add new features. Additionally, almost all machine learning algorithms contain *hyperparameters*, which are not learned during the training process but set by the user. Even the choice of loss function can be considered as an adjustable hyperparameter. There is also no free lunch in machine learning, since no single method works best over all possible scenarios [22]. Thus in order to choose the best model, some sort of model evaluation criteria needs to be established.

Evaluating the performance requires metrics that can be used to compare models, which again depend on e.g. whether we are dealing with a classification or regression problem. In classification, the predicted label always either matches the true label or not, while in regression the prediction is almost always wrong at least to some extent.

In classification problems accuracy is the most simple metric. It is defined as

$$\text{Accuracy} = \frac{\#\{\hat{y} = y\}}{n}, \tag{2.4}$$

where $\#\{\hat{y} = y\}$ is the number of correctly classified samples and n is the total number of samples.

We can also inspect the class-specific accuracy visually with a *confusion matrix*, which has both the predicted and true class cross-tabulated. From the confusion matrix, we can easily see how the predictions are distributed and if there is any imbalance between classes, which can cause the accuracy to be a misleading metric. If, for example, 90 percent of the samples are of the same class, then by always predicting that class we get a classifier with accuracy of 90 percent, which is however worthless in practice. Thus especially if the classes are imbalanced, we cannot rely on accuracy as the sole metric.

Luckily, there are also classification metrics that are not as heavily affected by class imbalance. Precision and recall are two metrics that are closely related to statistical concept of type I and type II errors. Precision of a class is defined as

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{2.5}$$

where TP (true positives) is the number of samples that were correctly labeled to that class and FP (false positives) is the number of samples that were incorrectly assigned to that class. Recall of a class, on the other hand, is defined as

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{2.6}$$

where FN (false negatives) is the number of samples that were incorrectly assigned to another class.

In regression, one commonly used metric is the mean squared error or MSE. It is defined as the mean of the squared loss (2.2)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2.7}$$

where n is the number of samples. An alternative to MSE is mean absolute error or MAE, which indicates the average absolute difference between the predicted and true output

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{2.8}$$

Chapter 3

Neural networks

Neural networks are a family of supervised machine learning methods, whose design was originally loosely inspired by the human brain. However, in the development of modern neural networks this neuroscientific approach has largely been abandoned and instead the field relies heavily on mathematics, especially calculus, linear algebra and probability, statistics and computer science [14]. While neural networks have been around for over 50 years, they have only risen to prominence during the last decade or so. This is, among other things, due to increased computational power and larger datasets becoming available [14]. Nowadays neural networks are used to tackle various kinds of problems, e.g. in image and speech recognition [17] [15].

In addition to sources cited in respective parts of the text, the overall structure and content of this chapter was influenced by Kallonen [26], who had a similar theoretical framework regarding neural networks in his thesis.

3.1 Feedforward neural networks

Feedforward neural networks are the most classic neural network models. They are called feedforward because of the way information 'flows' through the network without any cycles or loops [14].

A feedforward neural network consists of layers of *neurons*. The first layer which takes in the data is called *input layer* and the last layer which produces the final value is called *output layer*, whilst the intermediate layers are called *hidden layers*. The simplest possible network has only input and output layer, but in most cases a network has at least one hidden layer as well. Networks that have multiple hidden layers are called *deep neural networks* and training of such networks *deep learning*.

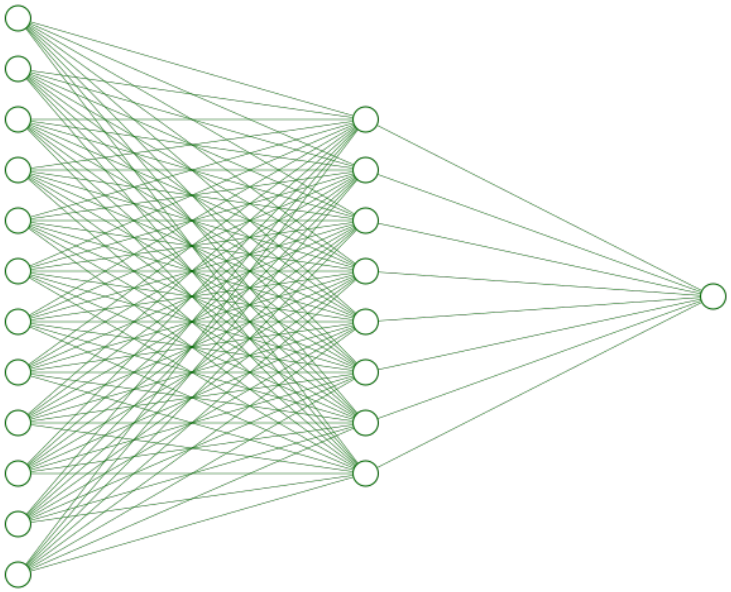

Figure 3.1: Structure of a dense feedforward neural network visualized. The network consists of an input layer with 12 neurons, one for each feature in the input vector, a sole hidden layer with 8 neurons and a output layer with one neuron. This particular network has 125 parameters, one for each neuron and connecting edge, emphasizing the fact that even for seemingly simple networks the number of parameters can be quite high compared to traditional statistical models, such as linear regression. Figure created with [30].

Illustrated by Figure 3.1, in *dense* networks all neurons on adjacent layers are connected through learnable weight parameters. A single neuron i calculates the weighted sum of inputs, which can be notated as

$$f(\mathbf{x}_i; \mathbf{w}_i, b_i) = \phi(\mathbf{x}_i^T \mathbf{w}_i + b_i),$$

where $\mathbf{x}_i$ is the vector of inputs, $\mathbf{w}_i$ is the weight vector, b_i is a bias term attached to each neuron and ϕ is an *activation function*. Activation function decides if a neuron 'fires', mimicking how biological neurons inside human brain work. Two widely used activation functions are rectified linear unit or ReLU and the softmax function. The latter, along with its binary counterpart sigmoid, is especially used in the output layer in classification tasks, whereas ReLu is common in the hidden layers. ReLU is defined as

$$\text{ReLU}(x) = \max(0, x), \tag{3.1}$$

while softmax is defined as

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{k} e^{x_i}}, \tag{3.2}$$

where k is the number of classes and $\mathbf{x} = (x_1, \ldots, x_k)$ is the vector of inputs. The values outputted by softmax sum up to 1 and can therefore be interpreted as probabilities. Activation function is called *linear*, if it does not perform any transformation. The main advantage of this approach is that the output values are not limited to any interval, which is particularly useful in regression tasks.

3.2 Training

Neural networks are trained using gradient-based optimization and the backpropagation algorithm. The core idea is to derive the gradient of the loss function with respect to weight parameters. The layered structure of neural network allows the gradient to be calculated using the chain rule of differentiation, which is what the backpropagation algorithm essentially does [16].

If we have a network with randomly initialized weight parameters $\mathbf{w}$ and a convex loss function $\mathcal{L}(y, f(\mathbf{x}; \mathbf{w}))$, by moving against the gradient $\nabla_{\mathbf{w}} \mathcal{L}(y, f(\mathbf{x}; \mathbf{w}))$ the loss function decreases, since a function always decreases fastest to the direction of the negative gradient. More mathematically expressed we update the weights $\mathbf{w}$ by

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}(y, f(\mathbf{x}; \mathbf{w})),$$

where α is a hyperparameter called learning rate, which determines how much weights are updated on each iteration. It can either be an immutable constant or a variable that is adjusted between iterations. In modern deep learning, the latter approach is more popular. One example of an adaptive learning rate algorithm is Adam [28], which estimates the first and second moments of the gradient to tune the learning rate.

The weights are updated until the loss function converges or pre-defined stopping criteria is met. In general, the method described above is called *gradient descent* because of the way the value of the loss function descends towards the local minimum. When deep learning is used in practice, for computational reasons the gradient is almost never calculated using the whole training data, but using a smaller randomly selected mini-batch instead. This approach is further called *stochastic gradient descent*.

3.3 One-dimensional convolution

Convolutional neural networks are neural networks that are designed for inputs with spatial or temporal grid-like structure [14] [37]. In the learning process, they utilize special mathematical operation called *convolution*. Currently, two-dimensional convolutional neural networks are the state-of-the-art method for image recognition. Below, we will however focus on one-dimensional convolutional networks, which have been particularly successful in tasks where the input is a time series sequence [29], such as the course of events in a ice hockey game.

In one-dimensional convolution a feature detector, also called *kernel*, with trainable parameters slides through the input vector, building up the next layer in the network with element-wise multiplication and summation. The height of the kernel and the step size, also called stride length, can be varied and thus thought as hyperparameters [37].

For more mathematical explanation, consider an input vector $\mathbf{i} = (i_1, \ldots, i_n) \in \mathbb{R}^n$ and a kernel $\mathbf{k} = (k_1, \ldots, k_m) \in \mathbb{R}^m$. If the stride length is one, the element j in the output or *feature map* $\mathbf{o} = (o_1, \ldots, o_{n-m+1}) \in \mathbb{R}^{n-m+1}$ is produced with

$$o_j = (\sum_{l=1}^{m} i_{j+l-1} \cdot k_l) + b,$$

where b is a bias term attached to kernel $\mathbf{k}$. One-dimensional convolution can also be applied when the input $\mathbf{i}$ is a matrix with size $n \times p$ instead of a one-dimensional vector. In this case, the kernel is also a matrix with p columns and the calculation performed is

$$o_j = (\sum_{l=1}^{m} \sum_{h=1}^{p} i_{j+l-1,h} \cdot k_{l,h}) + b.$$

One individual kernel can detect one pattern at different locations in the input. Because of this, when applied in practice, networks with one-dimensional convolutional layers almost always have multiple kernels, each building their own feature map. This is illustrated in Figure 3.2.

Convolutional networks are trained in similar fashion to regular feedforward networks. Since a kernel uses the same parameters in each step, convolutional networks generally require fewer parameters than fully-connected feedforward networks to learn equally complex patterns in the data, although they in most cases contain also standard dense layers. This parameter sharing also makes them much less prone to overfitting, because each kernel is fed with the whole input [37].

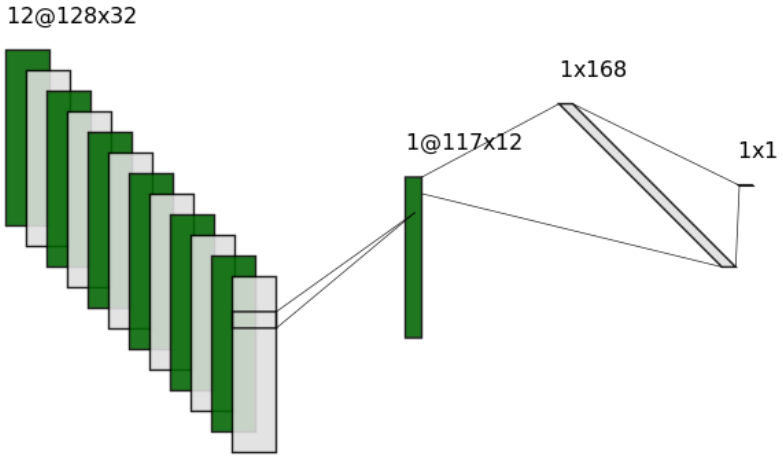

Figure 3.2: Structure of a one-dimensional convolutional network visualized. The first layer consists of 12 kernels of size 12×32 and takes 128×32 matrix as an input. With stride length of one, each of the 12 kernels slides down 128-12+1=117 steps to produce a feature map of size 117×12. Then a pooling layer with pooling region size 8×1 is applied to the feature map, condensating it to a flattened vector with 168 elements. This vector is the input to a standard dense layer, which outputs the final value. Figure created with [30].

We can further decrease the number of parameters in the network by reducing the size of feature maps. By applying *pooling*, we can condensate a strip in the feature map to a single summary statistic, such as the mean or maximum of that strip. This makes the network more resistant to small fluctuations in the input data [14].

3.4 Regularization

Since neural networks are quite complex by nature and can easily contain thousands of parameters, they require regularization in order to generalize properly. However, finding the optimal structure is not a trivial task of reducing the number of layers and neurons, because in almost all real-life scenarios the best performing model is relatively large [14].

One very simple regularization technique is *dropout*, where we randomly deactivate some proportion of the neurons in hidden layers. In *label smoothing*, we add small amount of noise to true labels with the intent of preventing the network becoming too confident with its predictions. Both methods have been shown to improve the performance of neural networks in wide variety of tasks [45] [35].

Chapter 4

Sports analytics

The use of statistical methods in team sports is a continuously evolving field of study. Sports data are collected and analyzed for a few main reasons. First and foremost, teams are eager to gain competitive advantage in a highly contested environment by all suitable means. Currently, every team in the National Basketball Association (NBA), the top basketball league in the world, has an analytics department [36]. Correspondingly in the NHL, an app was developed by Apple, SAP and the league itself to give coaches improved access to real-time statistics during games [11].

Another major utilizer of sports data is the sports betting industry. To set the odds, bookies use forecast models built on historical data to predict the outcome of a game. In a way, the whole business is based on the hypothesis of these predictions being more accurate in the long term than those of an average bettor. A study by Perez et al. [43] suggests this is the case at least for Spanish football.

Statistics are also consumed and studied by the fans without any commercial purpose. For instance, sports broadcasters typically provide fans with interesting and relevant statistical trivia during games. There are also dedicated third-party websites that make sports data freely available in an easily digestible format, such as Elite Prospects [1] and Hockey Reference [2] in ice hockey.

In this chapter, we will briefly present what kind of predictive analytics has been performed across different team sports. For the NHL specifically, we will also introduce a proposed upper bound for outcome prediction accuracy.

4.1 Outcome prediction

Predicting the outcome of a game and understanding the factors affecting it have long been questions of particular interest in various team sports. In this quest, vastly different methods and set of predictors have been tried through the years. In 2017, Lopez et al. [31] conducted a cross-sport study to understand how often the stronger wins in the four major North American professional leagues, the NHL, NBA, National Football League (NFL) and Major League Baseball (MLB). They used betting market data to build a Bayesian state-space model, which was used to estimate team strength levels between-season, within-season and game-to-game as well as home advantage. They concluded that the NBA and NFL show less random effect in outcome than the MLB and NHL, as team strength levels vary significantly less in the latter leagues.

A set of statistics measuring team quality can also be used as features. In his thesis, Weissbock [49] combined traditional statistics with more advanced performance metrics and pre-game textual reports by experts, which were analysed for sentiment, to predict the outcome of an NHL game with four different machine learning models. He was able to predict the winner of a single game with accuracy of 60 percent, whereas for a best-of-seven playoff series he reached accuracy of almost 75 percent using the combination of 30 different statistics.

Instead of predicting the outcome with information available before a game, there have also been studies where in-game win probability for each team at an arbitrary point in a game has been estimated. Robberechts et al. [44] used team strength in the form of Elo rating [19] combined with seven in-game statistics, such as the current score differential and number of yellow cards received by each team, to estimate in-game result probability in association football. Fitting a a hierarchical stochastic model on data from top tiers of English, Spanish, German, Italian, French, Dutch and Belgian football, their estimated probabilities for each outcome (home win, tie, away win) turned out to be well-calibrated when compared to historical performance of teams in the same situation.

The outcome can also be predicted indirectly by estimating the goal or score differential or point spread in a game. This approach is more popular in high scoring sports, such as basketball [25] and American football [7], but it has been used in low scoring association football [46] as well. Gelman [13] argues that even if the aim was to predict the outcome, score differential should be modeled instead, since it provides more information especially in games where the final margin is very narrow or wide.

4.2 Theoretical upper bound for prediction accuracy

Interestingly, Weissbock [49] could not obtain accuracy higher than 60 percent in single NHL game prediction regardless of the features used, implying that random chance affects the outcome. This led him to study whether an upper bound exists in outcome prediction accuracy. In his experiment, Weissbock used a Monte Carlo simulation, described by Algorithm 1 and similar to method used by Burke [8] earlier in NFL prediction, to simulate 10,000 seasons with varying levels of luck. Comparing the single season win percentages to single season win percentages of NHL teams in 7 seasons from 2005-06 through 2011-12, he suggested that the NHL is most similar to a league where 76% of the games are determined by luck and only 24% by skill. From this it follows that the upper bound for single game outcome prediction accuracy is $24\% + \frac{76\%}{2} = 62\%$.

if *rand(1, 100)* $\leq$ *luck* **then**
  winner = coin flip;
else
  winner = stronger team;
end

Algorithm 1: Algorithm used by Weissbock to determine the winner of a single game. Before each simulated season, the teams were assigned random strength levels with no team having the same strength.

Chapter 5

Data

5.1 Content

The data used in this thesis was originally extracted from https://www.nhl.com/ by Martin Ellis and released on Kaggle [12]. The data covers each of the 11,434 NHL games from season 2010-11 through 2018-19 and consists of nine separate CSV files, described in Table 5.1. For unknown reasons play-by-play data `game_plays` is not available for 190 games, but fortunately this proportion is not significant by any means. These games were ignored in the prediction.

In our study we will be focusing on using the sequence of in-game events, which is also called play-by-play data, provided by `game_plays` to predict the outcome of a game included in `game`. It is however worth noting that structure and richness of the data offers basis for building more complex models, e.g. taking the quality of team or even individual players into account in predictions.

5.2 Variables affecting prediction

As noted in Chapter 2, determining the variables used in the model is a vital part of machine learning process. There are 20 columns in the play-by-play data `game_plays`, showed in Table 5.2.

Even at a first glance it is obvious that most of the columns should not be used as features. The first five columns in the table are identifiers. They do not have any predictive power, but we need `team_id_for` to determine which team made the play and `game_id` to join the play-by-play data with the right game. If we wanted, we could also use `team_id_for` and `team_id_against` to include recent form, e.g. record in last 10 games, for each team,

Table 5.1: Data files and content

Name	Description
game	Basic information on each game.
game_goalie_stats	Statistics for each goaltender used in a single game.
game_plays	Every play of every game.
game_plays_players	Players involved in a play.
game_shifts	Information on which players were on ice at certain time.
game_skater_stats	Statistics for each player that played in a game.
game_teams_stats	Statistics for each team in a game.
player_info	Basic information on each player.
team_info	Basic information on each team.

which could be an useful feature. In this thesis however, we will only use the in-game data to predict the outcome.

The column `event` recognizes 22 types of in-game events, which are listed in Table 5.3. For someone familiar with ice hockey many of the events seem irrelevant for predicting the outcome of a game. The events from `Period Start` to `Early Intermission Start` are only related to arrangements of a game and can therefore be ignored, since we are more interested to examine the flow of the game. `Emergency Goaltender` occurs when either of the two regular goaltenders becomes unavailable on a short notice and a team needs to call-up an amateur goaltender on a emergency basis. This could be a significant factor affecting the result, but since there have only been three cases through the nine seasons and only one case of emergency goaltender actually playing in a game, we cannot reliably model the effect the event might have. The nine most common events from `Faceoff` to `Penalty` except for `Stoppage` are all actual in-game events and are thus used as features. `Stoppage` is omitted because each stoppage of play is followed by a face off, which means we do not lose any essential information by dropping it.

The column `secondaryType` provides a more specific subtype for `event`, e.g. shot type or particular foul resulting in a penalty. This information could perhaps improve the predictive accuracy, but including it would also increase the hierarchy of the model and require more feature engineering. In addition to this, it is reasonable to assume that the measurements for shot type are noisy, since there is not a clear-cut definition for e.g. a snap shot. For this reason, we choose to exclude it.

Rather than using raw x and y coordinates provided by `x` and `y`, we prefer to use adjusted

Table 5.2: Columns in `game_plays`. The bolded columns are included as features.

Name	Description
play_id	Unique identifier for each in-game event
game_id	Unique identifier for each game
play_num	Ordinal number of the in-game event
team_id_for	Unique identifier for the team making the play
team_id_against	Unique identifier for the team the play was made against
event	Type of the in-game event, e.g. shot
secondaryType	Subtype of the in-game event, e.g. wrist shot
x	x coordinate of the in-game event (unit foot)
y	y coordinate of the in-game event (unit foot)
period	Ordinal number of the period
periodType	Indicates whether the period is regulation, overtime or shootout
periodTime	Time in seconds from start of the period
periodTimeRemaining	Time in seconds remaining in the period
dateTime	Timestamp
goals_away	Number of away goals at the time of the in-game event
goals_home	Number of home goals at the time of the in-game event
description	More specific description of the event
st_x	x coordinate adjusted to be for attacking left to right (unit foot)
st_y	y coordinate adjusted to be for attacking left to right (unit foot)
rink_side	side of the rink from the NHL official's perspective

values in `st_x` and `st_y`, which correspond to position relative to opponent goal. Based on manual inspection these measurements seem to be relatively accurate. The columns from `period` to `dateTime` can be ignored, since the in-game events are in chronological order and the exact timestamp of an event is irrelevant, as in this thesis we are not for example aggregating events from multiple games. The columns `goals_away` and `goals_home` are excluded, since they are directly related to the outcome and final goal differential and hence using them as features would make the models pointless. The column `description` provides information on which players were involved in a play, which could be used to e.g. model probability of a shot from a certain player from a certain part of the rink resulting in a goal. However, this is beyond the scope of this thesis. Lastly `rink_side` is a simple indicator with no meaningful relation to events on the ice.

Table 5.3: In-game events by type. The events above the red line except for `Stoppage` are used in prediction.

Event	Number of occurrences
Face off	674,022
Shot	626,037
Hit	531,273
~~Stoppage~~	~~526,985~~
Blocked Shot	324,663
Missed Shot	266,975
Giveaway	194,516
Takeaway	155,700
Penalty	92,745
Goal	64,210
Period Start	37,638
Period Ready	37,590
Period End	37,585
Period Official	37,584
Game End	11,244
Game Scheduled	11,240
Shootout Complete	1183
Official Challenge	871
Game Official	229
Early Intermission End	28
Early Intermission Start	28
Emergency Goaltender	3

5.3 Additional features

In this thesis we restrict to using the features presented in the previous section in prediction. However, one way to improve the expected prediction accuracy would be by creating new and more descriptive features based on the existing ones. For example, we could use the relative x and y coordinates `st_x` and `st_y` to determine what was the distance to goal for an event, which could be particularly useful for shots. As illustrated by Pellinen [42] in his thesis, probability of a shot resulting in a goal depends strongly on the distance. Figure 5.1 supports this. The effect is most likely due to the fact that shots taken closer to the goal are more likely to hit the target and also leave the goaltender with less reaction time on average.

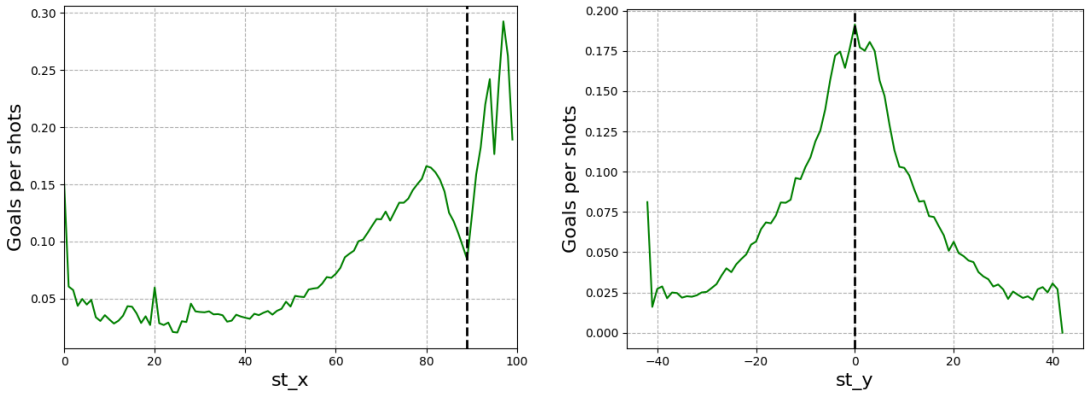

Figure 5.1: Proportion of goals per shots as a function of `st_x` (left) and `st_y` (right). The dashed black lines indicate the respective `st_x` and `st_y` coordinates of the net. The high scoring ratio for shots taken behind the goal line (`st_x` > 89) is likely explained by the fact that most attempts behind the net are not counted as shots, unless they end up in the net. Overall, only 475 goals have been scored behind the goal line, which amounts to less than one percent of the total number of goals in the play-by-play data.

Regarding shots another useful metric could be angle relative to goal, which we are also able to calculate based on `st_x` and `st_y`. Generally shots taken from center part of the ice are considered more dangerous than those taken from near the boards. From the coordinates and `team_id_for` we are also able to infer which team is spending more time at the offensive zone, which is also assumed to be an important factor in the outcome of a game.

5.4 Preprocessing

Play-by-play data in `game_plays` was prepared for modeling by first choosing only the relevant columns and in-game events, as discussed in section 5.2. Only the plays in regulation were included, as we are interested in predicting the outcome after 60 minutes and since in approximately 76.4 percent of the games only the three regulation periods are played. After each goal the game re-starts with a center ice face-off, which is a pattern a neural network could very well learn. However, we do not consider this desirable, since the pattern is more related to arrangements of a game. Thus all face-offs with `st_x` and `st_y` equaling zero were removed from the data. All 6,607 rows with missing data were also dropped, because proportion of such rows was only around 0.024 percent and dealing with missing values can be cumbersome.

After this, the games with no play-by-play data available were filtered out. Game data

Table 5.4: Data transformation process visualized. Most columns have been omitted for clarity.

team_id_for	team_id_against	event	st_x	st_y
NaN	NaN	Game Scheduled	NaN	NaN
NaN	NaN	Period Ready	NaN	NaN
NaN	NaN	Period Start	NaN	NaN
4.0	1.0	Faceoff	0.0	0.0
4.0	1.0	Giveaway	-28.0	-24.0

↓

home_team_for	event_blocked_shot	event_giveaway	event_dummy	st_x	st_y
1	0	1	0	-0.57	-1.01
1	1	0	0	-0.94	-1.18
⋮	⋮	⋮	⋮	⋮	⋮
0	0	0	1	0	0
0	0	0	1	0	0

game was then sorted by game_id and merged with game_plays based on the same column. The information provided by team_id_for and home_team_id on game was summarized to a dummy variable showing whether it was the home team that made the play. The output matrix $\mathbf{y}$, indicating which team won or if the game was tied after 60 minutes, was derived on the basis of column outcome, originally present on game. There is a minor imbalance between values of $\mathbf{y}$, as approximately 42.6 percent of the games resulted in a home win, 33.8 percent in an away win and 23.6 percent were tied after regulation. The column event was encoded as a set of dummy variables, each unique type of in-game event in addition to one new dummy event representing one column. Relative coordinates st_x and st_y were scaled by

$$\mathrm{st}(x) = \frac{x - \bar{x}}{s}, \tag{5.1}$$

where $\bar{x}$ is the mean and s the standard deviation of vector x, to have zero mean and unit variance. This standardization has been shown to speed up the convergence of gradient descent [21].

Following this, each game was split into its own two-dimensional array containing the in-game events in that game. The arrays were zero-padded to have as many rows as the game

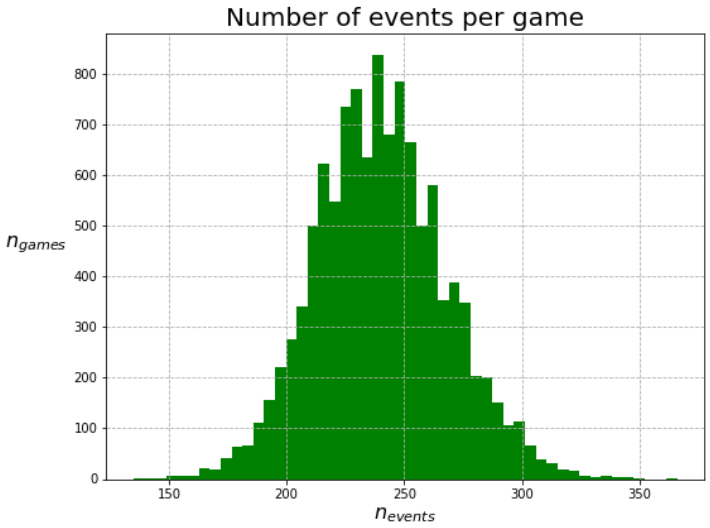

Figure 5.2: Number of events per game before zero padding.

with the most events, setting the value of the dummy event to one for the padded rows. After this each array had 366 events and 12 features. As illustrated by Figure 5.2, on average 100-150 rows were padded to each array.

Lastly, the two dimensional arrays were stacked to form the final three-dimensional input matrix $\mathbf{X}$ with shape $11244 \times 366 \times 12$. The column names of both $\mathbf{X}$ and $\mathbf{y}$ were also stored into a separate file.

5.5 Possible alterations

The structure of the preprocessing program was designed to enable a few modifications to both $\mathbf{X}$ and $\mathbf{y}$. As we are also interested in predicting the final goal differential, instead of $\mathbf{y}$ indicating the outcome, it can also be formulated as the goal differential after 60 minutes. In this case, $\mathbf{y}$ is a vector and not a matrix. Alternatively, $\mathbf{y}$ can be reduced to a binary variable by replacing ties with either home or away win, as indicated by `outcome`. Since approximately 54.6 percent of the games result in a home win and 45.4 percent in an away win, this would produce a more balanced class distribution. Additional features presented in section 5.3 can be included to the models. A separate program provides implementation for these features. The in-game events used as features can also be customized by the user.

Chapter 6

Methods

6.1 Technical implementation

The computational part of this thesis was conducted with Python. In data preprocessing Python libraries `NumPy` [39] and `pandas` [32] were the primary workhorses. The modeling part relied on Google's `TensorFlow` [3], which includes popular deep learning library `Keras` [9]. It provides implementation for all the neural network architectures introduced in Chapter 3, in addition to loss functions presented in section 2.2. General machine learning library `scikit-learn` [41] was also utilized both in the preprocessing and modeling stage. Data visualizations were created with `Matplotlib` [18] and `seaborn` [48]. The source code produced for the thesis is available on GitHub [27].

6.2 Models used

In this thesis we use neural network architectures introduced in Chapter 3 to predict the outcome and final goal differential of a game. As discussed in Chapter 3, a network with one-dimensional convolutional layers should be able to learn patterns in a time series sequence, such as the play-by-play data we are using. To examine this properly, a standard feedforward network was trained alongside a network with convolutional layers. Both neural network models were fitted with the preprocessed play-by-play data presented in Chapter 5. For both prediction tasks, the models had identical structure except for the output layer.

No formal model selection process was applied in this thesis, as it was not considered essential for the primary objective of this thesis. Thus the choice of hyperparameters, such as the number of layers and neurons, can be considered arbitrary for both networks. The neural network models were trained using Adam optimizer [28], with log-loss (2.3) being the loss

function for outcome classification model and mean squared error (2.7) for goal differential prediction. The number of epochs was set to 10. According to `Keras` [9] documentation, one epoch corresponds to each sample in the training data used once in the optimization of the loss function. All other parameters were set to `TensorFlow` version 2.0.0 defaults.

6.2.1 Feedforward

The feedforward model consists of four hidden layers. On the top there are two dense layers with 16 neurons each, both having ReLU (3.1) as the activation function. Then a flattening layer reshapes the input to one-dimensional format before a dropout layer randomly sets 50 percent of the neurons on the previous layer to zero. Lastly, for the outcome prediction model, output layer of three neurons produces the probabilities for each outcome using softmax (3.2) activation. By contrast, in goal differential prediction model the output layer consists of a single neuron with linear activation function. The structure of the feedforward network used in outcome classification is visualized in Figure A.1.

6.2.2 Convolution

The convolutional network has a slightly more complex structure. First there are two one-dimensional convolutional layers with 64 and 32 kernels respectively. Kernel length of 12 was used for the first layer and 8 for the second. They are followed by an average pooling layer which takes the mean of a 3×1 strip. After that the network has another set of two one-dimensional convolutional layers, each having 16 kernels and kernel length of 6. Then the inputs are summarized by global average pooling layer, which takes the mean of the whole input feature map, producing a one-dimensional vector which is fed to a dropout layer with a dropout rate of 0.5. Final layer is again a standard dense layer with three or one neurons, depending on the prediction task. Each convolutional layer has ReLU (3.1) as the activation function, while the output layer uses softmax activation (3.2) in outcome classification and linear activation in goal differential prediction. The layout of the convolutional network used to predict the outcome is visualized in Figure 6.1.

6.2.3 Reference points

When machine learning is used to solve a task, it is considered good practice to fit a simple model alongside the more sophisticated machine learning model. This is done to examine what is the additional value achieved by using a more complex method.

In outcome prediction multinomial logistic regression was used as a benchmark point, while in goal differential estimation classic linear regression was the reference model. Mathematical

Table 6.1: Summarized game statistics in `game_teams_stats` used as features in the reference point models. All statistics except `faceOffWinPercentage` were included for both teams.

Name	Description
shots	-
hits	-
pim	Penalty Infraction Minutes
powerPlayOpportunities	-
faceOffWinPercentage	-
giveaways	-
takeaways	-

formulations for these models are provided by Definitions 6.2 and 6.1, respectively. By contrast to neural network models which were fed with the play-by-play data, both reference point models were fitted with summarized game statistics. This was done to study if the spatial and temporal structure of the play-by-play data improves the quality of predictions. The game statistics, described in Table 6.1 were standardized (5.1) similarly to the relative coordinates in play-by-play data before training. For both models, default values provided by `scikit-learn` version 0.22 were used for all the parameters.

Definition 6.1 (Linear regression)**.** Given an input vector $\mathbf{x}_i = (1, x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^{p+1}$, the predicted output $\hat{y}_i$ for sample i is

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} = \mathbf{x}_i^T \boldsymbol{\beta}.$$

The weight vector $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p) \in \mathbb{R}^{p+1}$ is estimated by

$$\boldsymbol{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y},$$

where $\mathbf{X} \in \mathbb{R}^{n \times (p+1)}$ is the input data matrix and $\mathbf{y}$ is the vector of true observed values.

Definition 6.2 (Multinomial logistic regression)**.** Given an input vector $\mathbf{x}_i = (1, x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^{p+1}$, the probability that sample y_i belongs to class c is

$$P(y_i = c) = \frac{e^{\mathbf{x}_i^T \boldsymbol{\beta}_c}}{\sum_{j=1}^{K} e^{\mathbf{x}_i^T \boldsymbol{\beta}_j}},$$

where K is the number of classes. Here, the weight vector $\boldsymbol{\beta}_j$ is estimated similarly to linear regression for each class j.

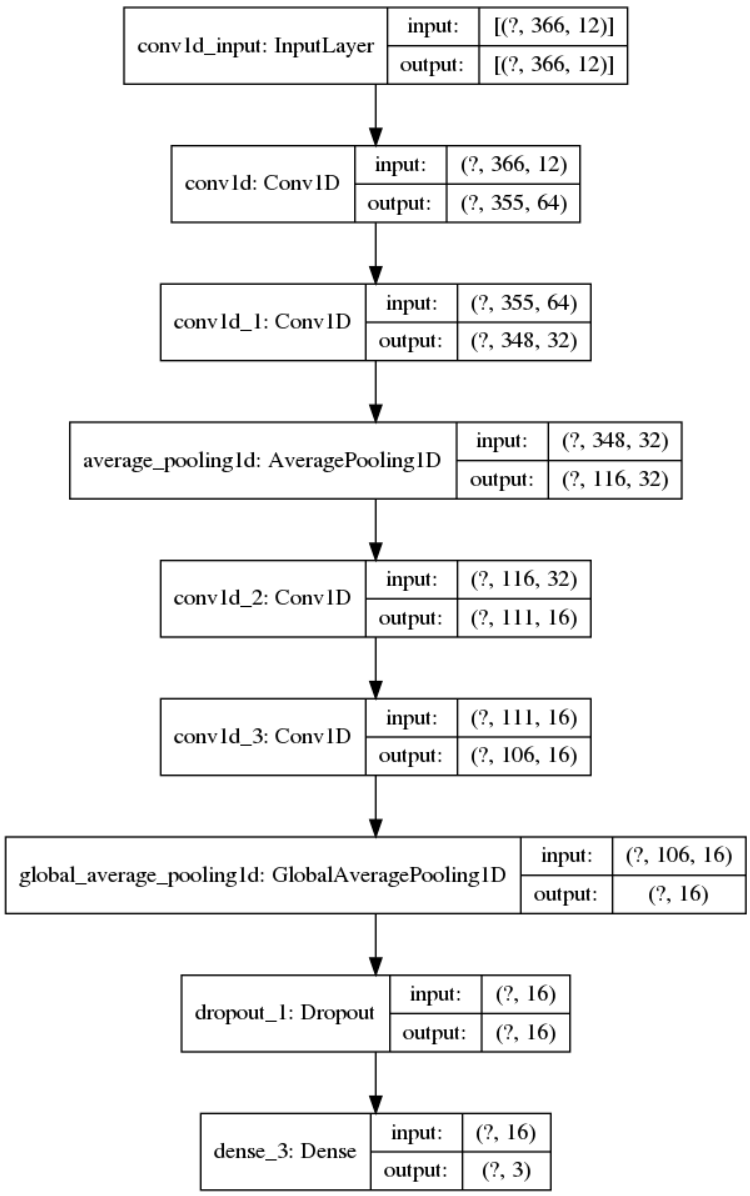

Figure 6.1: Convolutional network used for outcome prediction visualized. Question marks denote the number of samples or games, which does not affect the internal structure of the network. Figure created with `Keras` function `plot_model`.

Chapter 7

Results

In this chapter we will finally present the key findings of this thesis and put them into context. In section 7.1 we will show the results from the outcome classification model and in section 7.2 we will evaluate the performance of the goal differential prediction models. In section 7.2 we will also present a classifier that is produced by converting the estimated goal differentials to outcomes.

Before fitting the models introduced in Chapter 6, both input $\mathbf{X}$ and output $\mathbf{y}$ were randomly split to train and test data with 80:20 ratio, with 8,995 games used for training and the remaining 2,249 for testing. For both neural network models, the training data was further split to training and validation sets with the same 80:20 ratio.

7.1 Outcome

Table 7.1 reports the comparison of the three outcome classification models using common classification metrics calculated on the test data. A naive classifier always predicting home win (the majority class) with probability 1 was used as a baseline.

We can see that both neural network models, especially the one-dimensional convolutional network, perform better compared to the logistic regression model. Accuracy obtained with the neural network models is in line with 60% reported by Weissbock et al. [50], although they only predicted the winner in their study and based their predictions on information available before a game. The result is expected, as the neural networks are trained with the richer play-by-play data instead of summarized game statistics. In addition both neural networks have much more complex structure with more trainable parameters, allowing them to presumably learn more descriptive patterns in the data. By comparison, the logistic regression model has

Table 7.1: Classifiers compared. Here log-loss is the mean loss per sample and precision and recall are the unweighted means among all classes. For log-loss lower is better and for other metrics higher.

Model	Log-loss	Accuracy	Precision (average)	Recall (average)
Feedforward	0.967	0.576	**0.53**	0.52
Convolution	**0.844**	**0.616**	0.47	**0.54**
Logistic regression	0.987	0.520	0.51	0.47
Baseline	19.888	0.424	0.14	0.33

42 parameters, one for each of the 13 features per class plus intercept term for each class, whereas the feedforward network has 18,051 trainable parameters and the convolutional network 30,387.

As illustrated by Figure 7.1, there is a major imbalance between predicted classes for the neural network models, as neither of the two are able to predict ties at a proper rate. This could be considered somewhat unexpected, as the class distribution in the training data is fairly balanced. Still, it might be that the minor imbalance in class distribution causes the models to classify ties as wins to either side. It has been shown that even small class imbalance can affect the classification performance of a feedforward neural network [23]. A variety of methods have been suggested to address the issue [4], but applying these strategies in practice for our data is beyond the scope of this thesis.

7.2 Goal differential

Correspondingly, the results from the three goal differential models are presented in Table 7.2. A model always predicting the mean goal differential in the training data was used as a baseline. As with the classifiers, the neural network models outperform the linear regression and baseline models, with convolutional network producing both the lowest mean squared error and mean absolute error. Similarly to outcome prediction, this can likely be attributed to more adaptive structure of the former models, as well as richer training data.

By looking at Figure 7.2 we can see that both neural models underfit by failing to predict large absolute goal diffentials accurately. Especially the predictions of the feedforward network seem to be almost unaffected by the true goal differential. This could suggest that the models are not able to learn properly during the training phase or that they are too tightly regularized, which causes the networks to make 'safe' predictions with regard to the loss function. However, increasing the number of epochs to 100 causes the convolutional network to overfit

Table 7.2: Goal differential models compared on test data. Lower is better for both metrics.

Model	MSE	MAE
Feedforward	4.383	1.630
Convolution	**4.101**	**1.517**
Linear regression	4.615	1.709
Baseline	5.318	1.830

drastically on the training data, while the model's inability to predict larger goal differentials persists on the test data.

Since approximately 46% of NHL games are decided by less than two goals, it is uncertain whether any of the models is actually useful when predicting the outcome of a game. To test this, we can convert the goal differential to outcome by labeling games with predicted goal differential larger than 0.5 as home wins, games with goal differential less than -0.5 as away wins and games with goal differential 0 ± 0.5 as ties. By converting the goal differentials predicted by the convolutional network on the test data this way, we are able to obtain a classifier with accuracy of 0.607, unweighted mean precision of 0.57 and unweighted mean recall of 0.56. The precision and recall scores are actually slightly better than those produced by the convolutional network that predicted outcome directly, which supports the argument by Gelman [13]. Consequently, the outcomes predicted by this classifier are also more representative to the true outcome distribution, as illustrated by Figure A.2.

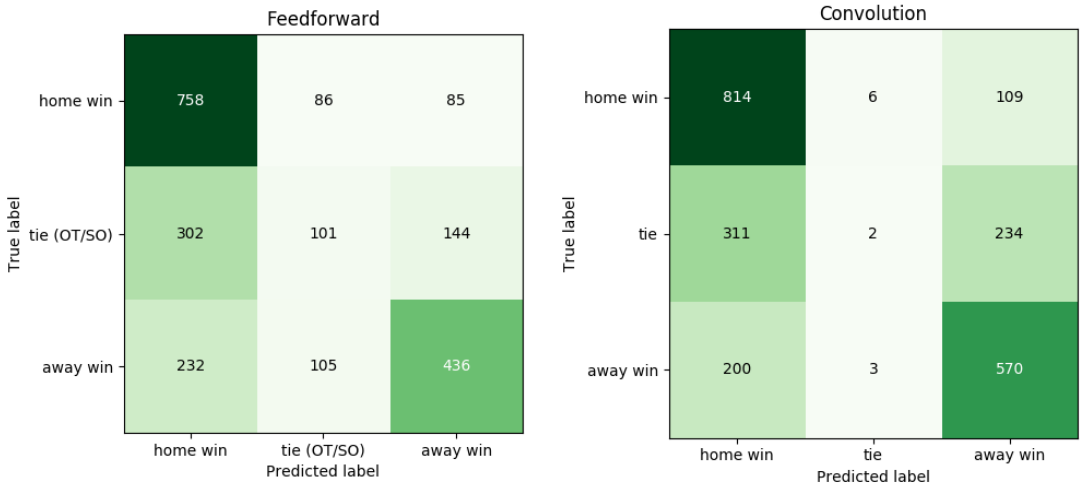

Figure 7.1: Confusion matrices for the outcome classification models.

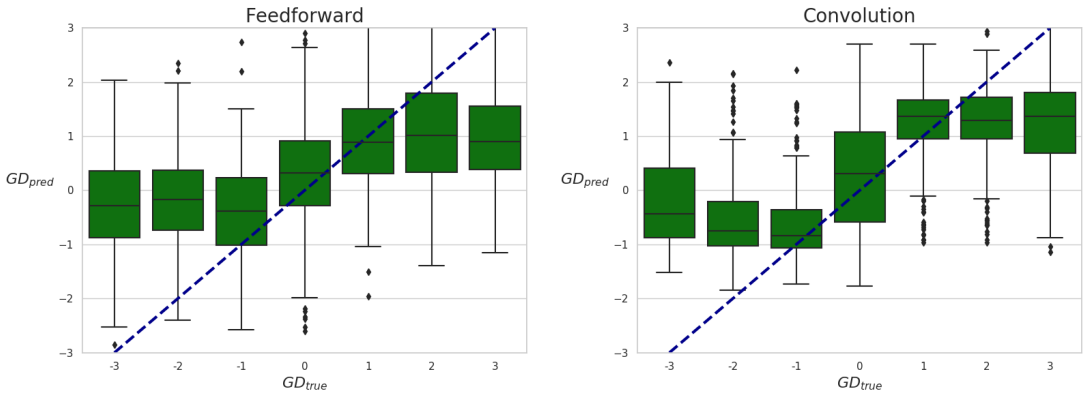

Figure 7.2: Distribution of predicted goal differentials per true goal differential on test data. For clarity, games with absolute true goal differential larger than three have been omitted. The dashed blue line represents cases where the predicted and true goal differentials are equal, while the black line inside each box is the median prediction for each true goal differential.

Chapter 8

Conclusion

In this Chapter we will conclude our work by discussing the most important results of this thesis. In addition to that, we will also introduce some ideas for future research on the topic.

8.1 Discussion

In Chapter 7 we discovered that the outcome classification models failed to predict ties at a proper rate, while the goal differential models were not able predict large absolute goal differentials accurately. Therefore the goodness of fit can be considered somewhat questionable and flawed in both prediction tasks. By applying a more formal model selection procedure the quality of fit could improve. For instance, automated hyperparameter tuning has been shown to provide better results for neural networks than manual search [5], which was the method used in this thesis. `Keras Tuner` [40] is a framework that offers hyperparameter tuning for models built with `Keras`. There are also other neural network architectures that could be tried on the data. Recurrent neural networks are a family of neural networks that have been developed for processing sequential data [14].

Additionally, incorporating new features, such as distance and angle of a shot as presented in section 5.3, to the models should enhance the results. Another possibly useful tweak to data could be regularization with label smoothing [35], which could be particularly well suited to ice hockey game outcomes. For example, games that are tied after three periods are typically games that could have resulted in a win for either team. Label smoothing would also allow separating wins by margin, smoothing the outcome classes more for wins by a small margin than those with a large final goal differential.

Although a properly calibrated model trained with richer data is likely to perform better,

the gap to be covered between the best model presented in this thesis and a model with perfect accuracy is quite large. This suggests that there could exist a upper bound for prediction accuracy, which further implies that we are not always able to identify the better team based on the course of events in a game.

The upper bound is not necessarily equal to 62 percent proposed by Weissbock [49], even though the prediction accuracy of 61.6% obtained in this thesis is almost identical to 59.8% reported by them. This is because in this thesis the better team and luck are defined differently, as the approach is also different. In Weissbock experiment, the team considered stronger before a game was also considered to be the better team. The team strength levels were fixed before each simulated season and consequently all games, where a stronger team was beaten by a weaker one, were credited to luck. By contrast, in this thesis this setting is not sensible, since we are determining the better team on an individual game level. In a league with high parity, such as the NHL, it is reasonable to assume that an underdog can beat a favorite in a single game simply by playing better without random chance having major effect on the outcome.

In the light of evidence found in this thesis it seems plausible that random chance affects the outcome of a game, as we were not able to infer the outcome with distinctly high accuracy even though we knew 'everything' that went on in a game. However, assessing the effect luck might have is difficult, since we do not have a good quantitative metric for it in single ice hockey game prediction. Moreover, the results support the argument that the final goal differential should be predicted instead of outcome, as converting the predicted goal differentials to outcomes produced a more balanced classifier.

8.2 Future work

There are multiple ways to continue research building upon the groundwork laid on this thesis. Probably the most obvious starting point for a future study would be to apply similar methodology to other team sports and compare the results to those obtained in this thesis. The play-by-play data required to perform the study is available for at least the NBA [6], although not for free. Comparison to the NBA could be particularly relevant, since Lopez et al. [31] concluded that the NBA is more predictable than NHL with information available before a game.

The neural network models could also be trained with events in the first two periods and then used to predict the outcome after 60 minutes in a way that is similar to Robberechts et al. [44] in principle. By doing so, it would be possible to study if games, for which the outcome is likely to change in the third period, can be identified based on the course of the game. The

play-by-play data could also be combined with pre-existing knowledge of team strength, if the goal was to construct a model that would predict the result of a single game as accurately as possible.

Alternatively, the estimation of random chance could be performed over multiple games instead of only one single game. The in-game events from a streak of games played by a team could be aggregated and then used to predict success of the team during that streak. One of the main advantages of this approach would be that the amount of data available for each prediction would be manifold compared to single game prediction. In this case, the motivation for a study could be to find out if the assumed effect of random chance is smaller when the sample size is larger. As noted in Chapter 4, Weissbock [49] was able to obtain more accurate results when predicting the winner of a best-of-seven playoff series.

Appendix A

Appendix

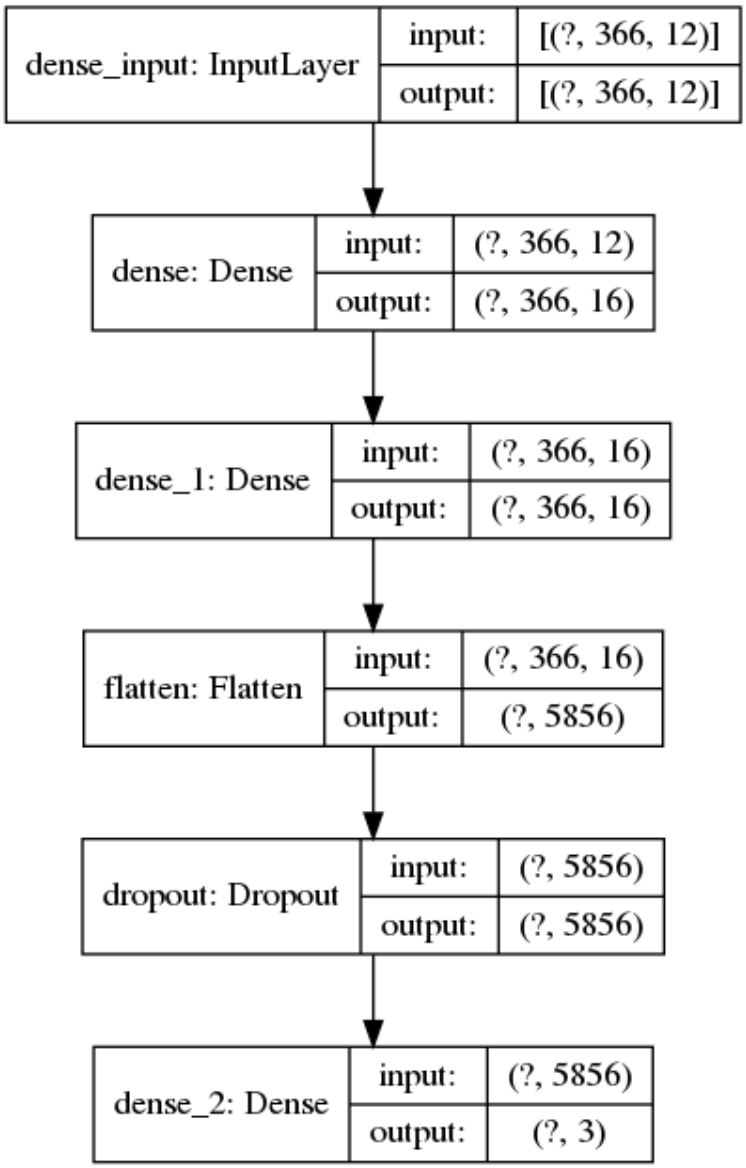

Figure A.1: Feedforward network used for outcome prediction visualized. Question marks denote the number of samples or games, which does not affect the structure of the network. Figure created with `Keras` function `plot_model`.

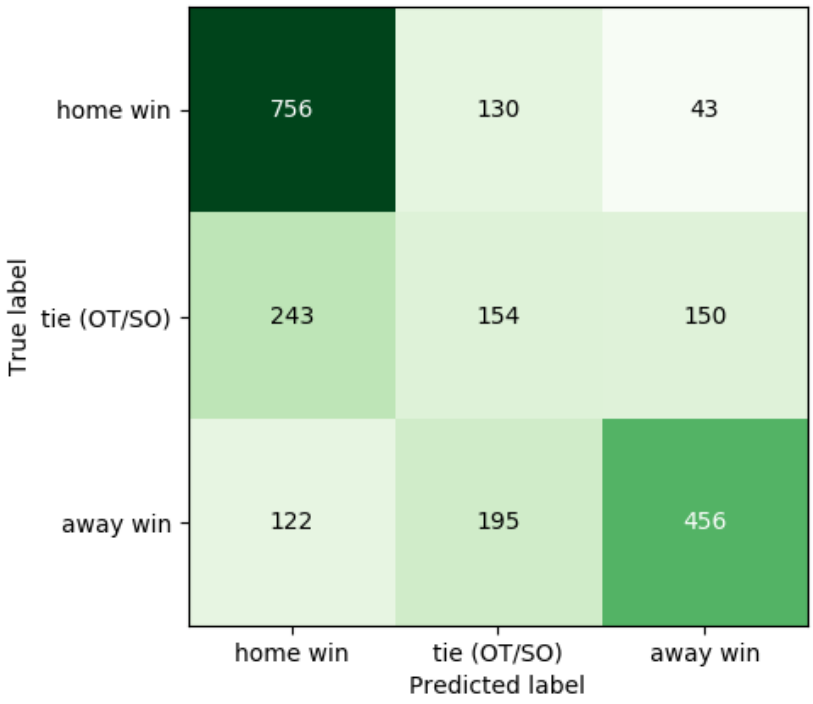

Figure A.2: Confusion matrix for the converted classifier.

Bibliography

[1] Elite Prospects. https://www.eliteprospects.com/. Last Accessed 20 Feb 2020.

[2] Hockey Reference. https://www.hockey-reference.com/. Last Accessed 20 Feb 2020.

[3] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from https://www.tensorflow.org.

[4] Aida Ali, Siti Mariyam Hj. Shamsuddin, and Anca L. Ralescu. Classification with class imbalance problem: a review. In *SOCO 2015*, 2015.

[5] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.

[6] BigDataBall. NBA Historical Play-By-Play Game Logs. https://www.bigdataball.com/nba-historical-playbyplay-dataset/. Last Accessed 19 March 2020.

[7] Andrew Blaikie, Gabriel Abud, and John A. David. NFL & NCAA Football Prediction using Artificial Neural Networks, 2011.

[8] Brian Burke. Luck and NFL Outcomes 3. http://archive.advancedfootballanalytics.com/2007/08/luck-and-nfl-outcomes-3.html. Last Accessed 13 March 2020.

[9] François Chollet et al. Keras. https://keras.io, 2015.

[10] Wikimedia Commons. Layout of a hockey rink. https://commons.wikimedia.org/wiki/File:Icehockeylayout.svg, 2006. Last Accessed 30 March 2020.

[11] Dan Rosen. https://www.nhl.com/news/new-app-will-give-nhl-coaches-real-time-stats-during-games/c-303980192. Last Accessed 20 Feb 2020.

[12] Martin Ellis. NHL Game Data. https://www.kaggle.com/martinellis/nhl-game-data, 2019.

[13] Andrew Gelman. Basketball Stats: Don't model the probability of win, model the expected score differential. https://statmodeling.stat.columbia.edu/2014/02/25/basketball-stats-dont-model-probability-win-model-expected-score-differential/. Last Accessed 5 March 2020.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[15] Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks, 2013.

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. Springer, 2 edition, 2009.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[18] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. Software available from https://matplotlib.org/3.1.0/index.html.

[19] Lars Magnus Hvattum and Halvard Arntzen. Using ELO ratings for match result prediction in association football. *International Journal of Forecasting*, 26(3):460 – 470, 2010.

[20] International Ice Hockey Federation. IIHF Rule Book 2018-22. https://iihfstorage.blob.core.windows.net/iihf-media/iihfmvc/media/downloads/rule%20book/iihf_official_rule_book_2018_ih_191114.pdf. Last Accessed 13 March 2020.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[22] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer, 2014.

[23] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. *Proceedings of the 2000 International Conference on Artificial Intelligence ICAI*, 06 2000.

[24] Jatkoajan keskustelupalsta. Jääkiekko ja sattuma. https://keskustelu.jatkoaika.com/threads/jaakiekko-ja-sattuma.59994/. Last Accessed 11 March 2020. In Finnish.

[25] Eric Jones. Predicting outcomes of NBA basketball games. Master's thesis, North Dakota State University of Agriculture and Applied Sciences, 2016.

[26] Kimmo Kallonen. Quantitative Comparison of Deep Neural Networks for Quark/Gluon Jet Discrimination. Master's thesis, University of Helsinki, 2019.

[27] Daniel Kari. nhl-game-prediction. https://github.com/danielkari95/nhl-game-prediction. Last Accessed 20 March 2020.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[29] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1D Convolutional Neural Networks and Applications: A Survey, 2019.

[30] Alex LeNail. Publication-ready NN-architecture schematics. http://alexlenail.me/NN-SVG/. Last Accessed 01 April 2020.

[31] Michael J. Lopez, Gregory J. Matthews, and Benjamin S. Baumer. How often does the best team win? A unified approach to understanding randomness in North American sport, 2017.

[32] Wes McKinney. Data Structures for Statistical Computing in Python, 2010. Software available from https://pandas.pydata.org/.

[33] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[34] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2nd edition, 2018.

[35] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help?, 2019.

[36] NBA Stuffer. https://www.nbastuffer.com/analytics101/nba-teams-that-have-analytics-department/. Last Accessed 20 Feb 2020.

[37] Michael A. Nielsen. Neural networks and deep learning, 2015.

[38] Nils J. Nilsson. Introduction to Machine Learning: An Early Draft of a Proposed Textbook, 1998.

[39] Travis Oliphant. NumPy: A guide to NumPy, 2006. Software available from https://www.numpy.org/.

[40] Tom O'Malley. Hyperparameter tuning with Keras Tuner. https://blog.tensorflow.org/2020/01/hyperparameter-tuning-with-keras-tuner.html. Last Accessed 9 March 2020.

[41] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Software available from https://scikit-learn.org/stable/.

[42] Jani Pellinen. Maalitodennäköisyyksien mallintaminen jääkiekossa. Master's thesis, University of Jyväskylä, 2019. in Finnish.

[43] Levi Perez, Jaume Garci, and Plácido Rodríguez. Forecasting football match results: Are the many smarter than the few?, 2016.

[44] Pieter Robberechts, Jan Van Haaren, and Jesse Davis. Who Will Win It? An In-game Win Probability Model for Football, 2019.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[46] Raymond Stefani. Predicting score difference versus score total in rugby and soccer. *Ima Journal of Management Mathematics - IMA J MANAG MATH*, 20:147–158, 08 2008.

[47] Matthew Stewart. The actual difference between statistics and machine learning. https://towardsdatascience.com/the-actual-difference-between-statistics-and-machine-learning-64b49f07ea3. Last Accessed 7 February 2020.

[48] Michael Waskom et al. seaborn: statistical data visualization. Software available from https://seaborn.pydata.org/.

[49] Joshua Weissbock. Forecasting Success in the National Hockey League using In-Game Statistics and Textual Data. Master's thesis, University of Ottawa, 2014.

[50] Joshua Weissbock, Herna Viktor, and Diana Inkpen. Use of Performance Metrics to Forecast Success in the National Hockey League. In *MLSA@PKDD/ECML*, 2013.